TA Session 6

Anthony Gillioz
Institute of Computer Science – University of Bern, Switzerland

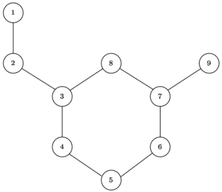
Contact: anthony.gillioz@unibe.ch

Theoretical Tasks

Task 1

1. Compute the following topological descriptors for the following unlabeled graph (the numbers on the nodes reflect an arbitrary node order index):

   - *First Zagreb index*,
   - *Narumi simple topological index*,
   - *Polarity number*,
   - *Wiener index*,
   - *Randic index*, and
   - *Balaban-J index*.

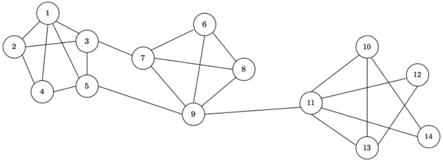


- You can use Python for this exercise.
- The *Balaban-J-Index* is dependent on the number of edges in the graph, denoted by m. Remember that for an undirected graph, each edge is counted twice (in the definition of the LNs).

Theoretical Tasks

Task 2

2. Compute the two unary features (i.e., the *leading eigenvalues* and the *eigen-mode volumes*) as well as the *Inter-mode adjacency matrix* of the following unlabeled graph (the numbers on the nodes reflect an arbitrary node order index):



- You can use Python for this exercise.

Theoretical Tasks

Task 3

3. Apply the Spanning prototype selector with $n = 3$ to a graph data set of five graphs $\mathcal{T} = \{g_1, \dots, g_5\}$. The distances between pairs of graphs $d_{ij} = d(g_i, g_j)$ are given by

$$\mathbf{D} = \begin{pmatrix} 0 & 1 & 3 & 7 & 11 \\ 1 & 0 & 2 & 9 & 8 \\ 3 & 2 & 0 & 6 & 14 \\ 7 & 9 & 6 & 0 & 3 \\ 11 & 8 & 14 & 3 & 0 \end{pmatrix}$$

- Describe the different steps that lead to your results.
- Report the prototype graph set $\mathcal{P}$.

Theoretical Tasks

Task 4

4. Using the Prototype set P you found in the previous exercise, embed all graphs $g \in \mathcal{T}$ in $\mathbb{R}^3$. Then compute the pairwise Euclidean distance in the embedding space and compare these distances to their actual distance in the original graph domain (using the distance matrix).

- You can use Python for this exercise.
- Report the graph embedding in $\mathbb{R}^3$, the pairwise Euclidean distance matrix.
- Compare the dissimilarity matrix $\mathbf{D}$ from the previous exercise with the pairwise Euclidean matrix you obtained.

Implementation Task

In this implementation task, you have to use a dissimilarity measure to embed graphs in a vector space.

Remarks:

- The entire code must be contained within the file `PR_lecture/Exercise_6/ex6.py`.
- You are allowed to modify the code as much as you want, including changing function signatures, creating new functions or classes, and so on.

Implementation Task

Create an $N \times N$ dissimilarity matrix $\mathbf{D} = (d_{ij})$ by applying a graph matching algorithm (viz. graph edit distance) to all pairs of graphs located in `PR_lecture/Exercise_6/graphs`. Then, use this dissimilarity matrix $\mathbf{D}$ to embed the N graphs into a 2D plane through MDS and generate a scatter plot to visualize them. Save the plot you obtained in `PR_lecture/Exercise_6/results/plot_mds.png`.

Implementation Task

Remarks:

- To compute the graph dissimilarity measure you can:
  1. use the graph edit distance algorithm you implemented in Series 2.
  2. use the graph edit distance algorithm provided in `networkx.algorithms.similarity`.
- If you use the GED from `networkx.algorithms.similarity` use only a small number of iterations.
- The cost associated with node and edge deletion, insertion and substitution are defined in `series6.pdf`.
- You find in `PR_lecture/Exercise_6/graphs/classes.csv` the class assigned to each graph. Each point on the MDS plot must be colored according to its corresponding class.

Implementation Task

Idea of code structure

```python
def main():
    # Code Here


if __name__ == '__main__':
    main()
```